Apple Assembly Line

$1.80

Volume 5 -- Issue 8 May, 1985

S-C Macro Assembler for ProDOS

At long last, the news you've all been waiting for: the ProDOS version of the S-C Macro Assembler is almost ready. We have a working assembler in Beta testing, and it's doing just fine We need to spend another month or two shaking on it and developing documentation, so it will be just a little longer 'til we start shipping, but it's on the way! Watch the front page of AAL for the announcement.

News from Don Lancaster

After nearly a year of delay at the publisher, Enhancing Your Apple II and //e, volume 2 is here! This followup to his very popular collection of Apple tricks, gimmicks, and techniques contains still more high-quality information on how to get the most out of our favorite computer. Here Don provides the tricks of microjustification and proportional spacing for Applewriter //e, an absolute "Old Monitor" style RESET for the //e, a software-only video synchronization technique for all Apple II's and //e's, and a just-for-fun guide to mapping and playing Castle Wolfenstein.

I've been saving the best for last: Tearing Into Applewriter //e. Here is 86 pages of priceless data on the internal workings of the most popular Apple Word Processor, including how to capture source code and customize it to your own taste. See our ad on page three for price and shipping.

As you will notice from his ad in this issue, Lancaster has been hard at work tearing into Appleworks, and has a set of disks available on that program. We haven't seen those yet, but I'll bet they're more of the same great inside info we've come to expect from Don.

A New Catalog for DOS 3.3 .....

Robert F. O'Brien
Dublin, Ireland

In AAL March '85 Bob S-C presented re-writes of some loosely coded DOS sections to make space for patches - the Catalog Function Handler is another such loose bit of code, but rather than just free up some bytes I decided to add some useful features which Apple omitted and correct an annoying error at the same time. This new routine adds the following features to the CATALOG command:

1. Displays the free space remaining on the disk.

2. Allows you to terminate the Catalog during the normal pause after a screenful of files have been displayed by pressing the <ESC>-key (or other designated key).

3. Displays the correct number of sectors for each file in the Catalog for even the very large files - where the number of sectors exceeds 255 (which was the limit of the old PRINT.DECIMAL subroutine at $AE42 in DOS 3.3).

4. Optionally displays two filenames on each line of the Catalog - this is an 80-Column card option, also great for double-barrelled CATALOG printouts (for labels etc.).

In addition, the new Catalog retains the principal features of the old routine such as displaying the Volume number, the locked file indicator (*) and the file type abbreviation so that the user is not deprived of any essential information.

All the foregoing was achieved without using any additional DOS RAM space or zero-page locations other than that space already used by the Catalog Function Handler itself. Of course, something of the old routine had to be sacrificed in order to add the new features - it was necessary to omit the message "DISK VOLUME " from the beginning of the display. The 12-byte space where this message resided is now used to house a subroutine to check for locked files.

Even with all these enhancements, there are 17 free bytes left over! You could use some of them to print out an abbreviated form of the "DISK VOLUME " message, like "V=".

An additional constraint I saddled myself with in doing the re-write was that PRINT.DECIMAL (the DOS subroutine used to convert the hexadecimal numbers in locations $44,45 to decimal and print them) should retain its normal entry point ($AE42) so that the new code would be compatible with other programs which might use it.

For those who wish to get double-barrelled Catalog listings on an 80-column card or on a printer just change the "SEC" at line 2010 to "LSR". In other words, $AE12:4A will enable the wide printout, and $AE12:38 will put it back to normal.

S-C Macro Assembler Version 2.0.....$100
Version 2.0 Upgrade Kit for 1.0/1.1/1.2 owners.....$20
Source Code for Version 1.1 (on two disk sides).....$100
Full Screen Editor for S-C Macro (with complete source code).....$49
S-C Cross Reference Utility (without source code).....$20
S-C Cross Reference Utility (with complete source code).....$50
DISASM Dis-Assembler (RAK-Ware).....$30
Source Code for DISASM.....additional $30
S-C Word Processor (with complete source code).....$50
DP18 Source and Object.....$50
Double Precision Floating Point for Applesoft (with source code).....$50
S-C Documentor (complete commented source code of Applesoft ROMs).....$50
Source Code of //e CX & F8 ROMs on disk.....$15

(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

AAL Quarterly Disks.....each $15, or any four for $45

Each disk contains the source code from three issues of AAL, saving you lots of typing and testing.		Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
	1980	-	-	-	1
	1981	2	3	4	5
	1982	6	7	8	9
	1983	10	11	12	13
	1984	14	15	16	17
	1985	18			

AWIIe Toolkit (Don Lancaster, Synergetics).....$39
ES-CAPE: Extended S-C Applesoft Program Editor (new price, was $60) $40
"Bag of Tricks", Worth & Lechner, with diskette.....($39.95) $36

Blank Diskettes (Verbatim)..... package of 20 for $32
(Premium quality, single-sided, double density, with hub rings)
Vinyl disk pages, 6"x8.5", hold two disks each.....10 for $6
Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each
(Cardboard folders designed to fit 6"X9" Envelopes.) or $25 per 100
Envelopes for Diskette Mailers..... 6 cents each

quikLoader EPROM System (SCRG).....($179) $170
PROmGRAMER (SCRG).....($149.50) $140
D MAnual Controller (SCRG).....($90) $85
Switch-a-Slot (SCRG).....($190) $175
Extend-a-Slot (SCRG).....($35) $32

"Apple ProDOS: Advanced Features for programmers", Little..($17.95) $17
"Inside the Apple //c", Little.....($19.95) $18
"Inside the Apple //e", Little.....($19.95) $18
"Apple II+/IIe Troubleshooting & Repair Guide", Brenner.....($19.95) $18
"Apple][Circuit Description", Gayler.....($22.95) $21
"Understanding the Apple II", Sather.....($22.95) $21
"Understanding the Apple //e", Sather.....($24.95) $23
"Enhancing Your Apple II, vol. 1", Lancaster.....($15.95) $15
"Enhancing Your Apple II, vol. 2", Lancaster.....($17.95) $17
"Assembly Cookbook for the Apple II/IIe", Lancaster.....($21.95) $20
"Beneath Apple DOS", Worth & Lechner.....($19.95) $18
"Beneath Apple ProDOS", Worth & Lechner.....($19.95) $18
"6502 Assembly Language Programming", Leventhal.....($18.95) $18
"6502 Subroutines", Leventhal.....($18.95) $18
"Real Time Programming -- Neglected Topics", Foster.....($9.95) $9
"Microcomputer Graphics", Myers.....($12.95) $12
"Assem. Language for Applesoft Programmers", Finley & Myers.($16.95) $16
"Assembly Lines -- the Book", Wagner.....($19.95) $18

Add $1.50 per book for US shipping. Foreign orders add postage needed.

Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 ***
*** We accept Master Card, VISA and American Express ***

To install the new patches just BLOAD the two binary files: NEW CATALOG PART 1. and NEW CATALOG PART 2. You can put the modified DOS onto any normal disk using Bob Perkins' technique (in AAL Aug 1982 p.24) without disturbing any other files present. or INIT a blank disk and the modified DOS will be incorporated on it. If you prefer to terminate long Catalog's with the <RETURN> key as you do for listings with the S-C Macro Assemblers just change byte $AE21:8D.

Also, if you are prepared to restrict yourself to 11 character file-names you can have a double-barrelled Catalog on the 40-Column screen by changing byte $ADF7:0B (POKE 44535,11), but I feel it would be of little value overall.

Now for a more detailed look at the program internals. Due to the requirement to save as many bytes as possible to squeeze in the desired features it was not possible to write the code in as straight-forward a manner as one would like. Even so, the routine was written with 17 byte to spare - after many re-writes to fit in all the features.

Lines 1020-1240 define various subroutines, variables, and data tables inside the rest of DOS.

Lines 1320-1360 use the same code as the original Catalog routine to initialize the File Manager and read the disk Volume Table Of Contents (VTOC).

In lines 1370-1410 we clear LINE.SKIP.FLAG which is used by SKIP.LINE subroutine to determine whether to tab to a second column or print a carriage return. Then we call PRINT.DECIMAL.YA to print the volume number. The volume number itself is passed in the A-register, and a zero high-byte in Y. Since we stripped out the code for printing "DISK VOLUME ", the volume number will be printed immediately to the right of the CATALOG command, on the same line. You will see "CATALOG 254 395", or the like, where the first number is the volume number and the second is the count of free sectors.

By making a special entry above the PRINT.DECIMAL subroutine which is used both here and at line 1830 below, we save several bytes. Of course we have already save a couple dozen bytes by not printing "DISK VOLUME ".

Calculation of the free disk space is made in lines 1420-1530. We make use of a new feature in the corrected PRINT.DECIMAL routine whereby $44 and $45 are reduced to 0 during the conversion - resulting in a saving of 4 bytes by not having to re-zero $45. (In the old routine only $44 was reduced to 0.)

In the VTOC 4 bytes are set aside for each track to indicate sector usage although only 2 are needed for a standard Apple disk. (The extra space allows up to 50 tracks and up to 32 sectors per track to be initialized.) A bit set=1 means that the corresponding sector on the track is available for use. If a bit is set=0 then the sector is already allocated. So it was simply a matter of counting every bit set from offset byte $38 (track 0) to Byte $C3 (for Trk $22) of the VTOC buffer to get a

count of the free space. If you want to count all the way to the 50th track, in case the program is working with a hard disk like the Sider or Corvus, or a RANA 320K floppy, change lines 1430-1440:

```
1430        LDX #$38
1440        LDA VTOC.BUFFER,X
```

In line 1550 we have another departure from the original code - 2 bytes were saved by entering the tail end of the SKIP.LINE subroutine in order to set the number of lines to place on the screen before pausing during a Catalog. This has the added advantage that you can customize your Catalog more easily in that the line count can be adjusted by modifying a single byte ($AE25).

At lines 1570-1610 we start by clearing the Carry flag so that the first sector of the directory will be read (track $11, sector $0F). Also we set the index (X) to the first filename entry in the sector.

Lines 1620-1660 examine the track number of the Track/Sector list for the current filename entry. Should this number be 0 it indicates that we are at the end of the directory, at which point we would terminate the Catalog by exiting the File Manager routine by a jump to $B37F.

Fortunately, there was a JMP $B37F instruction within relative branching distance of the Catalog Function Handler. We could therefore dispense with the JMP to $B37F in the original code saving a further 3 bytes by branching to FM.EXIT at $AD86 instead. This is an address in the DELETE Function Handler ($AD2B-AD97) which precedes the Catalog routine in RAM. There are three ways we can terminate the Catalog, which all result in a branch to FM.EXIT: here at line 1600 when we find there are no more catalog sectors, at line 1650 when we find there are no more catalog entries, and at line 2090 when the ESC-key is typed during a screen-pause.

At line 1660 if the track number value is negative (bit 7 set) then we have found a deleted file. Deleted files don't show up on the Catalog, so we call on the subroutine at $B230 which sets the X-Register to the value of the entry point offset for the next entry in the sector, if any.

If on return from this subroutine the Carry flag bit is set (=1) then we have reached the end of the current catalog sector and we branch back to READ.SECTOR at line 1580 to read the next directory sector, if any. (Each directory sector accommodates 7 entries.)

At line 1680 we call the SKIP.LINE subroutine, which normally merely prints a carriage return. This routine was called from five different places in the original catalog code, so we have saved a dozen bytes by only calling it from this one place. (Putting it in-line would save four more!)

At line 1700 we call the new subroutine at the site of the DISK

VOLUME message space to check for locked files and print the space or asterisk. This routine also leaves the file type code in the Y-register This code could be placed in-line, rather than making it a subroutine, but then the final two lines could not be used as a short PRINT.SPACE subroutine.

Lines 1710-1790 convert the file type code to a file type character. The file type code is in bits 6-0, and is either zero (meaning type T), or a single bit. The hex values 40, 20, 10, 08, 04, 02, and 01 stand for file types B, A, R, S, B, A, and I. A string at $B4C8 holds "TIABSRAB", so we need to convert the bit position to an index value, and pick up the character out of that string. The ASL at line 1740 elminates the "lock/unlock" bit. The loop in line 1750-1770 shifts bits out until the value is zero, counting up in the Y-register. If the value was already zero, we exit immediately with Y=0, and type is "T". A type value of 1 gives an index of 1, up through $40 giving an index of 7.

By the way, types 40 and 20 are not Binary and Applesoft. They are hardly ever used, except in protection schemes. Types 04 and 02 are Binary and Applesoft.

The original catalog code had a significantly longer loop for converting the file type number to an index. You might want to compare the two.

The number of sectors in the file is picked up and converted in lines 1800-1830 and the decimal value is printed, surrounded by spaces. Lines 1840-1900 print out the file name.

Lines 1920-1950 advance to the next filename entry, and branch either to process it or to read in another catalog sector.

Lines 1970-2130 usually print a carriage return. If you have changed line 2010 to "LSR", to get double column catalogs, the least significant bit of LINE.SKIP.FLAG will determine whether to print a carriage return or not. When line 2010 is "SEC" we will always get a carriage return. If a carriage return is printed, we also count the line. When the line count is complete, we pause and wait for a keystroke. If that key is an ESC-key, the catalog will terminate. If not, the line count is re-initialized and we go back for more file names.

Line 2150 simply reserves 17 bytes, shoving the PRINT.DECIMAL routine down so that it still starts at $AE42 like it used to. These 17 bytes could be used for other code or data, whatever you like.

Lines 2160-2230 store a value to be converted and printed, print a blank, and then fall into the PRINT.DECIMAL subroutine.

The new corrected PRINT.DECIMAL subroutine is actually a little shorter than th buggy original. It left room for a JMP PRINT.SPACE at the end, which saved calling PRINT.SPACE from several other places. It also left room for the LINE.SKIP.FLAG variable.

The PRINT.DECIMAL subroutine (lines 2240-2490) effectively divides the number in $44,45 by subtracting in turn the values 100, 10 and 1 from it - a 16-bit subtraction. The count of the number of subtractions and the low order byte remainder are temporarily stored on the stack to conserve memory usage. We start with 100 and keep subtracting it and incrementing the subtraction-counter until we get borrow, at which point we print the counter value.

Now $44,45 will contain the remainder and so we continue using 10 and then 1 until three decimal digits are printed. This subroutine can accurately convert numbers having values up to 999 decimal.

CHALLENGE. Even though we have already squeezed out 17 bytes, while adding new features, we did lose the "DISK VOLUME " message. Can someone out there squeeze enough more out, without losing any features, to slip the message back in?

CAVEAT. If you decide to put this new CATALOG program on your disks, please be careful. There are some programs which temporarily patch the catalog routine themselves. In particular, ES-CAPE and other commercial programs patch the SKIP.LINES subroutine so that the pause is eliminated. Since SKIP.LINES has been moved and is different. no telling what might happen.

```
                    1000 *SAVE S.NEW CATALOG
                    1010 *-------------------------------
44-                 1020 DOS.ARITH.REG          .EQ $44,45
                    1030 *-------------------------------
B230-               1040 ADV.NEXT.DIR.ENTRY     .EQ $B230
ABDC-               1050 DOS.INIT.FM            .EQ $ABDC
AD86-               1060 EXIT.FM                .EQ $AD86
FDED-               1070 MON.COUT               .EQ $FDED
FD8E-               1080 MON.CROUT              .EQ $FD8E
FD0C-               1090 MON.RDKEY              .EQ $FD0C
B011-               1100 READ.DIRECTORY.SECTOR  .EQ $B011
AFF7-               1110 READ.VTOC              .EQ $AFF7
                    1120 *-------------------------------
B3A4-               1130 DEC.CONVERSION.TABLE   .EQ $B3A4
B3A7-               1140 FILE.TYPE.NAME.TABLE   .EQ $B3A7
                    1150 *-------------------------------
B39D-               1160 CATALOG.LINE.COUNT     .EQ $B39D
B4C6-               1170 DIRECTORY.ENTRY        .EQ $B4C6
B39C-               1180 DIRECTORY.INDEX        .EQ $B39C
B7F6-               1190 DISK.VOL.NUMBER        .EQ $B7F6
B4C9-               1200 FILE.NAME              .EQ $B4C9
B4E7-               1210 FILE.SIZE              .EQ $B4E7
B4C8-               1220 FILE.TYPE              .EQ $B4C8
B5F9-               1230 FM.VOL.NUMBER          .EQ $B5F9
B3BB-               1240 VTOC.BUFFER            .EQ $B3BB
                    1250 *-------------------------------
                    1260 *    New Catalog for DOS 3.3
                    1270 *    by  Robert F.O'Brien
                    1280 *-------------------------------
                    1290       .OR $AD98
                    1300       .TF NEW CATALOG PART 1
                    1310 *-------------------------------
                    1320 CATALOG
AD98- 20 DC AB      1330       JSR DOS.INIT.FM   Init file manager.
AD9B- A9 FF         1340       LDA #$FF          Set Volume = 0
AD9D- 8D F9 B5      1350       STA FM.VOL.NUMBER   (matches any volume)
ADA0- 20 F7 AF      1360       JSR READ.VTOC     Load in VTOC into buffer.
                    1370 *---Print Volume Number----------
ADA3- A0 00         1380       LDY #0            High byte = 0
ADA5- 8C 69 AE      1390       STY LINE.SKIP.FLAG    (signal to skip)
ADA8- AD F6 B7      1400       LDA DISK.VOL.NUMBER   low byte
ADAB- 20 3B AE      1410       JSR PRINT.DECIMAL.YA
                    1420 *---Calculate Free Space---------
ADAE- A2 74         1430       LDX #$74          Trk 0 VTOC offset
ADB0- BD 7F B3      1440 .1    LDA VTOC.BUFFER+$38-$74,X     Bit Map Byte
ADB3- 10 06         1450 .2    BPL .3            This sector in use
ADB5- E6 44         1460       INC DOS.ARITH.REG   Count a free sector.
ADB7- D0 02         1470       BNE .3
ADB9- E6 45         1480       INC DOS.ARITH.REG+1
ADBB- 0A            1490 .3    ASL               Check next bit
ADBC- D0 F5         1500       BNE .2            Still more in this byte
ADBE- E8            1510 .4    INX               Next byte of bit map
ADBF- D0 EF         1520       BNE .1            ...still more
ADC1- 20 42 AE      1530       JSR PRINT.DECIMAL    print number free
                    1540 *---Start Line count-------------
ADC4- 20 24 AE      1550       JSR SET.LINE.COUNT        lines to print.
                    1560 *---Start reading directory------
ADC7- 18            1570       CLC               Get first sector.
                    1580 READ.SECTOR
ADC8- 20 11 B0      1590       JSR READ.DIRECTORY.SECTOR
ADCB- B0 B9         1600       BCS EXIT.FM       No more sectors
ADCD- A2 00         1610       LDX #0            Index to 1st file in sector
                    1620 SET.ENTRY.INDEX
ADCF- 8E 9C B3      1630       STX DIRECTORY.INDEX   Set entry offset
ADD2- BD C6 B4      1640       LDA DIRECTORY.ENTRY,X See if valid filename
ADD5- F0 AF         1650 EXIT  BEQ EXIT.FM       ...end of directory
ADD7- 30 29         1660       BMI NEXT.ENTRY    ...ignore deleted file.
                    1670 *---Start next file display------
ADD9- 20 09 AE      1680       JSR SKIP.LINE     Next line or Tab
                    1690 *---Locked or Unlocked-----------
ADDC- 20 AF B3      1700       JSR LOCKED.FILE.CHECK     "*" if locked file.
                    1710 *---File Type--------------------
ADDF- 98            1720       TYA               Get file type byte
ADE0- A0 FF         1730       LDY #-1           Index to type table
ADE2- 0A            1740       ASL               Ignore Bit 7
```

```
ADE3- C8       1750 .1     INY                 Next file type code
ADE4- 4A       1760        LSR                 Check bit of type byte
ADE5- D0 FC    1770        BNE .1              ...not yet
ADE7- B9 A7 B3 1780 .2     LDA FILE.TYPE.NAME.TABLE,Y Get file type
ADEA- 20 ED FD 1790        JSR MON.COUT        ...and print it
               1800 *---File Size--------------------
ADED- BC E8 B4 1810        LDY FILE.SIZE+1.X          high order byte
ADF0- BD E7 B4 1820        LDA FILE.SIZE,X            low order byte
ADF3- 20 3B AE 1830        JSR PRINT.DECIMAL.YA       print total sect.
               1840 *---File Name--------------------
ADF6- A0 1E    1850        LDY #30
ADF8- BD C9 B4 1860 .3     LDA FILE.NAME.X            char. no. in Y.
ADFB- 20 ED FD 1870        JSR MON.COUT          print file name.
ADFE- E8       1880        INX                        next char.
ADFF- 88       1890        DEY
AE00- D0 F6    1900        BNE .3                     not done yet!
               1910 *---Next File in Directory-------
               1920 NEXT.ENTRY
AE02- 20 30 B2 1930        JSR ADV.NEXT.DIR.ENTRY  Set X-Reg for next file
AE05- 90 C8    1940        BCC SET.ENTRY.INDEX     more in sector
AE07- B0 BF    1950        BCS READ.SECTOR         get next sector
               1960 *-------------------------------
               1970 SKIP.LINE
AE09- 20 B6 B3 1980        JSR PRINT.SPACE   Separate 2nd line entry
AE0C- EE 69 AE 1990        INC LINE.SKIP.FLAG  Toggle lsbit
AE0F- AD 69 AE 2000        LDA LINE.SKIP.FLAG  Check Odd or Even
AE12- 38       2010        SEC         <<<Change to "LSR" for double
               2020 *                       column CATALOG >>>
AE13- 90 14    2030        BCC RETURN
AE15- 20 8E FD 2040        JSR MON.CROUT     Start a new line
AE18- CE 9D B3 2050        DEC CATALOG.LINE.COUNT    continue countdown
AE1B- D0 0C    2060        BNE RETURN                not full screen yet
AE1D- 20 0C FD 2070        JSR MON.RDKEY     Pause for keypress!
AE20- C9 9B    2080        CMP #$9B          Is it  ESC-key?
AE22- F0 B1    2090        BEQ EXIT          ...yes, exit file manager
               2100 SET.LINE.COUNT
AE24- A9 15    2110        LDA #21           lines per screenful
AE26- 8D 9D B3 2120        STA CATALOG.LINE.COUNT
AE29- 60       2130 RETURN RTS               Continue catalog
               2140 *-------------------------------
AE2A-          2150        .BS 17            17 FREE BYTES!
               2160 *-------------------------------
               2170 *   Print (YA) with leading and
               2180 *      trailing blanks.
               2190 *-------------------------------
               2200 PRINT.DECIMAL.YA
AE3B- 84 45    2210        STY DOS.ARITH.REG+1
AE3D- 85 44    2220        STA DOS.ARITH.REG
AE3F- 20 B6 B3 2230        JSR PRINT.SPACE
               2240 *-------------------------------
               2250 *   Print ($44,45) with trailing blank
               2260 *-------------------------------
               2270 PRINT.DECIMAL
AE42- A0 02    2280        LDY #2            Set for 3 divisors
AE44- A9 B0    2290 .1     LDA #$B0          ASCII zero
AE46- 48       2300 .2     PHA               save digit on stack
AE47- 38       2310        SEC               Subtract 100, 10. or 1
AE48- A5 44    2320        LDA DOS.ARITH.REG    from remainder
AE4A- F9 A4 B3 2330        SBC DEC.CONVERSION.TABLE,Y
AE4D- 48       2340        PHA               save remainder on stack
AE4E- A5 45    2350        LDA DOS.ARITH.REG+1
AE50- E9 00    2360        SBC #0            (divisor high byte = 0)
AE52- 90 0A    2370        BCC .3            ...far enough
AE54- 85 45    2380        STA DOS.ARITH.REG+1   Update remainder
AE56- 68       2390        PLA
AE57- 85 44    2400        STA DOS.ARITH.REG
AE59- 68       2410        PLA               get current digit
AE5A- 69 00    2420        ADC #0            and count the subtraction
AE5C- D0 E8    2430        BNE .2            ...continue subtracting
AE5E- 68       2440 .3     PLA               Discard stacked remainder byte
AE5F- 68       2450        PLA               Get quotient digit
AE60- 20 ED FD 2460        JSR MON.COUT      and print it!
AE63- 88       2470        DEY               Next divisor
AE64- 10 DE    2480        BPL .1            ...not finished yet
AE66- 4C B6 B3 2490        JMP PRINT.SPACE   Trailing space
               2500 *-------------------------------
AE69- 00       2510 LINE.SKIP.FLAG .DA #0    LEAST SIGNIFICANT BIT IS FLAG
               2520 *-------------------------------
```

```
                    2530         .OR $B3AF
                    2540         .TF NEW CATALOG PART 2
                    2550 *-------------------------------
                    2560 *   OVERLAYS "DISK VOLUME " MESSAGE
                    2570 *-------------------------------
                    2580 LOCKED.FILE.CHECK
B3AF- A9 AA         2590         LDA #"*"
B3B1- BC C8 B4      2600         LDY FILE.TYPE,X   file type code.
B3B4- 30 02         2610         BMI CAT.COUT      ...the file is locked
                    2620 PRINT.SPACE
B3B6- A9 A0         2630         LDA #" "
                    2640 CAT.COUT
B3B8- 4C ED FD      2650         JMP MON.COUT
                    2660 *-------------------------------
```

80-Column Window Utility for //e and //c.....Bill Reed
New Orleans, LA

I throughly enjoyed "Fast Text Windows" by Michael Ching. However, I prefer not to use the area at $800-BFF as a text buffer; I much prefer to use the first bank of the language card, which is not normally used by Applesoft programs running under DOS 3 3

I modified Mike's program by changing the immediate values in lines 1560 and 1580 from #$0C to #$D4 and adding lines 1644, 1646 and 1905. The first two lines enable the bank of RAM to be read or written to. The last re-enables the Applesoft ROMS.

```
1644        LDA $C08B
1646        LDA $C08B
1905        LDA $C082
```

I further modified the program to function in 80 columns on a //e or //c. The big problem was to mimic the text card, which uses bank switching to store adjacent characters in the same address, but different locations (main RAM and aux RAM). This was solved by using one buffer for the "even" characters and another for the "odd".

Additional code was required to determine the even/odd condition, so I (being lazy) removed the border portion of the program to conserve room. The border routines could certainly be retained if part of the program was also moved to bank one of the language card area (Be careful if you try this, because you must avoid calling the monitor or Applesoft ROMs when the ROMs are switched off. You can possibly get away with calling the monitor with the ROMs switched off, but only if you first make a copy of the monitor in the F800-FFFF area of RAM.)

I moved the data storage to the zero page, mostly because it was available and slightly faster.

```
                    1000 *SAVES.WINDOWS.80
                    1010 *-AAL APRIL 85 P 16-------------
                    1020         .OR $2F5
                    1030         .TF B.WINDOWS.80
                    1040 *-------------------------------
00-                 1050 TOP     .EQ $00  $0-$6 DATA
01-                 1060 BOTTOM  .EQ $01
02-                 1070 LEFT    .EQ $02
03-                 1080 RIGHT   .EQ $03
04-                 1090 WIDTH   .EQ $04
05-                 1100 LINE    .EQ $05
06-                 1110 DIREC   .EQ $06
                    1120 *
```

Why Are Apple Owners So Loyal?

People who have the best often are, but in the case of Apple there's more. Apple owners think back to how Apple got started in 1977, just two people working out of a garage and what happened is the talk of Wall Street and the computer industry as well. Many like the fact that Apple only makes computers. Unlike their competition, they don't make typewriters, copiers or telephones. They do just one thing and that's one reason they do it so well.

At Applied Engineering we think the same way. You see, Applied Engineering is the only major hardware manufacturer totally dedicated to the Apple computer. Whereas most of our competitors must divide their customer support and engineering time between IBM, Atari, Radio Shack or other computers, our engineers only design products for the Apple. This dedication allows us to be much more familiar with the Apple and those who use them.

We don't expect you to buy an Applied Engineering peripheral on loyalty alone, but when you compare our products to those made by QUADRAM, MICROSOFT, AST and others you'll find out why Applied Engineering means a quality design, innovation, craftsmanship and total Apple compatibility.

The other guys do pretty well considering how busy they are with IBM. But at Applied Engineering, ALL of our work involves the Apple. In fact, all of our employees were Apple owners before they came to work for us. The people in shipping, engineering, quality control, order entry, all use Apples at work and at home.

This one track mindedness of ours allows us to offer the largest storage with AppleWorks and Visicalc and our Z-80 card now includes the new 4.0 operating system. We can expand the Apple IIe to over 1 MEGABYTE of memory and we've got clock cards, music cards, A to D converters, digital controllers, and a BSR system so your Apple can control your whole house with no additional wiring!

Applied Engineering recognizes that we've got to do a better job than our IBM counterparts because we know you're smarter than the average computer buyer, you bought an Apple. You see, our competition has it a little easier, their customers aren't as smart as you. After all, they bought the wrong computer.

So if you need more memory, or 80 columns, or RGB color, double hi res graphics, if you want to know the time and temperature or other "real world" conditions, if you'd like to run CP/M software, 68000 software, have a RAM disk, increase the storage of AppleWorks or Visicalc, if you want your Apple to play music, talk and sing, if you'd like your Apple to control the lights and appliances in your house, then do what NASA does, what Ford does, what the U.S. Government, Hughes Aircraft, Honeywell, Westinghouse, AT&T, and even what Apple computer does. Call Applied Engineering. Then you will discover what thousands of companies and tens of thousands of Apple owners already know, that you can be smart and loyal all at the same time.

We Set the Standard

APPLIED ENGINEERING

(214) 492-2027

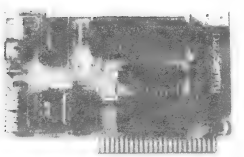
For information and specifications on Applied Engineering's line of Apple peripherals, please see our ads in this magazine. Prices are given.

APPLIED ENGINEERING

To a World of Compromise, We Make No Contribution

THE NEW TIMEMASTER II H.O.

- Absolutely, positively, totally PRO-DOS and DOS 3.3 compatible.
- Time in hours, minutes, seconds and milliseconds (the ONLY PRO-DOS compatible card with millisecond compatibility).
- 24 hour military format or 12 hour with AM/PM format.
- Date with year, month, day of week and leap year.
- Eight software controlled interrupts so you can run two programs at the same time (many examples are included).
- The only card recognized by both the DOS 3.3 and PRO-DOS versions of Apple Writer IIe.
- Appleworks will now time and date stamp all your data automatically.
- Compatible with ALL of Apple's languages. Many sample programs for machine code, Applesoft, CP/M and Pascal on 2 disks.
- On-board timer lets you time any interval up to 48 days long down to the nearest millisecond.
- Rechargeable nickle-cadmium battery will last over 20 years.
- Two BSR/serial ports for future expansion.



	PRO-DOS COMPATIBLE	INCLUDES DOS DATER	MILLISECOND TIME	YEAR DATA	LARGEST SAMPLE SOFTWARE	REMOTE SET PORT	BSR PORT	EMULATES ALL OTHER CLOCKS
H.O	YES	YES	YES	YES	YES	YES	YES	YES
BRAND A	NO	NO	NO	NO	NO	NO	NO	NO
BRAND T	NO	NO	NO	NO	NO	NO	NO	NO
BRAND M	NO	NO	NO	NO	NO	NO	NO	NO
BRAND P	NO	NO	NO	NO	NO	NO	NO	NO
BRAND C	NO	NO	NO	NO	NO	NO	NO	NO
BRAND T	NO	NO	NO	NO	NO	NO	NO	NO

Full emulation of all other clocks. Yes, we emulate Brand A, Brand T, Brand P, Brand C, Brand S and Brand M too. It's easy for the H.O. to emulate other clocks, we just drop off features. That's why the H.O. can emulate others, but none of the others emulate us. The Timemaster II H.O. will automatically emulate the correct clock card for the software you're using. You can also give the H.O. a simple command to tell it which clock to emulate. This is great for writing programs for those poor unfortunates who bought some other clock card.

REMOTE CONTROL

Our BSR X-10 interface option for the H.O. allows you to remotely control up to 16 lights and electrical appliances through your BSR X-10 home control system in your home or office. You're already wired because a BSR system sends its signals over regular 120 volt wiring. That means you can control any electrical device in your home or office without additional wiring.

PRICE $129.00 BSR Option (may be added later) $49.00

VIEWMASTER 80

There used to be about a dozen 80 column cards for the Apple. Now there is only ONE.

- TOTALLY Videx Compatible.
- 80 characters by 24 lines, with a sharp 7x9 dot matrix.
- On-board 40/80 soft video switch with manual 40 column override.
- Fully compatible with ALL Apple languages and software—there are NO exceptions.
- Low power consumption through the use of CMOS devices.
- All connections are made with standard video connectors.
- Both upper and lower characters are standard.
- All new design (using a new Microprocessor based C.R.T. controller) for a beautiful razor sharp display.
- The VIEWMASTER incorporates all the features of all other 80 column cards, plus many new improvements.

	PRICE	BUILT-IN SOFTSWITCH	SHIFT-KEY SUPPORT	100% POWER UP DESIGN	80 COLUMN HOME	7X9 DOT MATRIX	LIGHT PEN INPUTS	80 COLUMN OVERRIDE	INVERSE CHARACTERS
VIEWMASTER	139	YES	YES	YES	YES	YES	YES	YES	YES
[illegible]	MORE	NO	YES	NO	NO	NO	NO	YES	YES
[illegible]	MORE	NO	NO	NO	NO	NO	NO	YES	YES
[illegible]	MORE	NO	YES	NO	NO	YES	NO	YES	NO
[illegible]	MORE	NO	YES	NO	NO	YES	NO	YES	NO
VIEWMASTER 80	MORE	YES	YES	NO	NO	YES	NO	YES	YES
SMARTERM	MORE	NO	YES	NO	NO	YES	NO	YES	YES
[illegible]	MORE	NO	NO	NO	NO	NO	NO	YES	YES

The VIEWMASTER 80 works with all 80 column applications including CP/M, Pascal, WordStar, Format II, Easywriter, Apple Writer II, VisiCalc, and all others. The VIEWMASTER 80 is THE MOST compatible 80 column card you can buy at ANY price!

PRICE $139.00

Z-80 PLUS

Now Includes New 4.0™ Software

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering and introduce your Apple to thousands of new programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system for running CP/M programs ever. Only CP/AM 4.0™ has advanced features like built-in disk emulation for popular memory expansion boards (those made by Apple and Applied Engineering and others) to give you a faster system with more storage. You also get menu driven utilities that are much easier to use than the older CP/M utilities so you can get down to all that great CP/M software faster. If you already own the Z-80 Plus, you can upgrade to the 4.0 software for only $29. The Z-80 Plus runs older CP/M programs too, down to Version 1.6 (2.2 is the most popular). With the Z-80 Plus you can run the largest body of software in existence. Simply plug the Z-80 Plus into any slot in your Apple. You'll have two computers in one and the advantages of both, all at an unbelievably low price.

- TOTALLY compatible with ALL CP/M software.
- The only Z-80 card with a special 2K "CP/M detector" chip.
- Fully compatible with microsoft disks (no pre-boot required).
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the II+ and Franklin).
- Runs WORD STAR, dBASE II, TURBO PASCAL, FORTRAN-80, PEACHTREE and ALL other CP/M software with no pre-boot.
- A semi-custom I.C. and low parts count allows the Z-80 Plus to fly thru CP/M programs at a very low power level. (We use the Z-80A at a fast 4MHZ.)
- Does EVERYTHING the other Z-80 boards do, plus Z-80 interrupts.

PRICE $139.00

SUPER MUSIC SYNTHESIZER — END MOCKINGBOREDOM

- Complete 16 voice stereo music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away by inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.
- Now with new improved software for the easiest and the fastest music input system available anywhere.
- We gives you lots of software. In addition to Compose and Play programs, 2 disks are filled with over 30 songs ready to play.
- Easy to program in Basic to generate complex sound effects. Now your games can have explosions, phaser zaps, train whistles, death cries. You name it, this card can do it.
- Four white noise generators which are great for sound effects.
- Full control of attack, volume, decay, sustain and release.
- Our card will play notes from 30HZ to beyond human hearing.
- Automatic shutoff on power-up or if reset is pushed.
- Many many more features.
- Works in any slot of a IIe or II+ including slot 3 of a IIe.

PRICE $159.00

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products work in Apple IIe, II, II+ and Franklin. Applied Engineering also manufactures a full line of data acquisition and control products for the Apple: A/D converters and digital I/O cards, etc. Please call for more information. All our products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle **THREE YEAR WARRANTY.**

APPLIED ENGINEERING

Call (214) 492-2027, 9 a.m. to 11 p.m. 7 days a week or send check or money order to P. O. Box 798, Carrollton, Texas 75006. MasterCard, Visa and C.O.D. welcome. No extra charge for credit cards. Texas residents add 5¼% sales tax. Add $10.00 if outside U.S.A.

```
18-                    1130 B1         .EQ $18,19    TEXT PNTR
1A-                    1140 B2         .EQ $1A,1B    BUFR PNTR
1C-                    1150 B3         .EQ $1C,1D    BUFR PNTR
20-                    1160 WNDLFT     .EQ $20
21-                    1170 WNDWDTH    .EQ $21
22-                    1180 WNDTOP     .EQ $22
23-                    1190 WNDBTM     .EQ $23
28-                    1200 BASL       .EQ $28
29-                    1210 BASH       .EQ $29
                       1220 *--------------------------------
03F5-                  1230 AMPERV     .EQ $3F5
C054-                  1240 PAG2OFF    .EQ $C054   READ MRBRD
C055-                  1250 PAG2ONN    .EQ $C055   READ AUXBRD
C082-                  1260 LCROM      .EQ $C082
C08B-                  1270 LCRAM1     .EQ $C08B
E6F8-                  1280 GETBYTE    .EQ $E6F8
E74C-                  1290 COMBYTE    .EQ $E74C
FBC1-                  1300 BASCALC    .EQ $FBC1
FC58-                  1310 HOME       .EQ $FC58
                       1320 *--------------------------------
                       1330 SETUP
02F5- A9 00            1340         LDA #MOVE.WINDOW
02F7- 8D F6 03         1350         STA AMPERV+1
02FA- A9 03            1360         LDA /MOVE.WINDOW
02FC- 8D F7 03         1370         STA AMPERV+2
02FF- 60               1380         RTS
                       1390 *--------------------------------
                       1400 MOVE.WINDOW
0300- 20 F8 E6         1410         JSR GETBYTE
0303- 86 00            1420         STX TOP
0305- 86 05            1430         STX LINE
0307- 20 4C E7         1440         JSR COMBYTE
030A- 86 01            1450         STX BOTTOM
030C- 20 4C E7         1460         JSR COMBYTE
030F- 86 02            1470         STX LEFT
0311- 20 4C E7         1480         JSR COMBYTE
0314- 86 03            1490         STX RIGHT
0316- E8               1500         INX
0317- 38               1510         SEC
0318- 8A               1520         TXA
0319- E5 02            1530         SBC LEFT
031B- 85 04            1540         STA WIDTH
031D- 20 4C E7         1550         JSR COMBYTE
0320- CA               1560         DEX
0321- 86 06            1570         STX DIREC
                       1580 *--------------------------------
                       1590 *--------------------------------
                       1600 MOVE.LINE
0323- A5 05            1610         LDA LINE
0325- 20 C1 FB         1620         JSR BASCALC
0328- A5 29            1630         LDA BASH
032A- 85 19            1640         STA B1+1
032C- 49 D4            1650         EOR #$D4
032E- 85 1B            1660         STA B2+1
0330- 18               1670         CLC
0331- 69 04            1680         ADC #$04       2ND BUFR
0333- 85 1D            1690         STA B3+1
0335- A5 28            1700         LDA BASL
0337- 85 18            1710         STA B1
0339- 85 1A            1720         STA B2
033B- 85 1C            1730         STA B3
033D- AD 8B C0         1740         LDA LCRAM1     ENABLE LANG
0340- AD 8B C0         1750         LDA LCRAM1     CARD R/W
                       1760 *--MOVE THE LINE SEGMENT--------
0343- A5 03            1770         LDA RIGHT
0345- 4A               1780         LSR            A/2 + EVN/ODD
0346- A8               1790         TAY            TXT SCRN PNTR
0347- A6 06            1800         LDX DIREC
0349- D0 1B            1810         BNE .3
                       1820 *--MOVE IT UP-------------------
034B- A6 04            1830         LDX WIDTH      DOWN COUNTER
034D- 90 07            1840         BCC .2
034F- B1 18            1850 .1      LDA (B1),Y     DO ODD COLS
0351- 91 1A            1860         STA (B2),Y
0353- CA               1870         DEX
0354- 30 29            1880         BMI .6
0356- AD 55 C0         1890 .2      LDA PAG2ONN    DO EVN COLS
0359- B1 18            1900         LDA (B1),Y
035B- 91 1C            1910         STA (B3),Y
035D- AD 54 C0         1920         LDA PAG2OFF
```

```
0360- 88          1930        DEY
0361- CA          1940        DEX
0362- 10 EB       1950        BPL .1
0364- 30 19       1960        BMI .6
                  1970 *--MOVE IT DOWN----------------
0366- A6 04       1980 .3     LDX WIDTH
0368- 90 07       1990        BCC .5
036A- B1 1A       2000 .4     LDA (B2),Y    DO ODD COLS
036C- 91 18       2010        STA (B1),Y
036E- CA          2020        DEX
036F- 30 0E       2030        BMI .6
0371- AD 55 C0    2040 .5     LDA PAG2ONN   DO EVN COLS
0374- B1 1C       2050        LDA (B3),Y
0376- 91 18       2060        STA (B1),Y
0378- AD 54 C0    2070        LDA PAG2OFF
037B- 88          2080        DEY
037C- CA          2090        DEX
037D- 10 EB       2100        BPL .4
                  2110 *--NEXT LINE-------------------
037F- E6 05       2120 .6     INC LINE
0381- AD 82 C0    2130        LDA LCROM     RESTORE ROM
0384- A5 01       2140        LDA BOTTOM
0386- C5 05       2150        CMP LINE
0388- B0 99       2160        BCS MOVE.LINE
                  2170 *--IF CLEARING, SET WINDOW------
038A- A5 06       2180        LDA DIREC
038C- D0 15       2190        BNE .7
038E- A6 02       2200        LDX LEFT
0390- 86 20       2210        STX WNDLFT
0392- A6 04       2220        LDX WIDTH
0394- CA          2230        DEX
0395- 86 21       2240        STX WNDWDTH
0397- A6 00       2250        LDX TOP
0399- E8          2260        INX
039A- 86 22       2270        STX WNDTOP
039C- A6 01       2280        LDX BOTTOM
039E- 86 23       2290        STX WNDBTM
03A0- 20 58 FC    2300        JSR HOME
03A3- 60          2310 .7     RTS
                  2320 *-------------------------------
```

AUTO/MANUAL Toggle Update for .....Robert F. O'Brien
S-C Macro Assembler Version 2.0 . Dublin, Ireland

Here is a short routine (23 bytes) which makes use of the ESC-U command option to toggle the Auto-linenumbering mode on and off readily. The routine is relocatable so you can put it anywhere you have sufficient free space - just set the ESC-U vector to point to it, in this case :$C083 C083 D00C:4C 00 03 N C080.

When the cursor is waiting for input at the beginning of the command line, typing ESC-U will generate the command AUTO and then you have the option of entering a line number and/or RETURN. To cancel the AUTO mode just type ESC-U while the cursor is at the beginning of the line (just after the linenumber - 4 or 5 digit line numbers are catered for).

Extended AUTO command:

The second routine, starting at $317. is just 17 bytes long and extends the AUTO command so that you can specify the increment after the starting linenumber. For example, AUTO 3000,1 sets a starting line number of 3000 and an increment of 1. This code is also relocatable but you must patch the first instruction in the main AUTO command so that it uses the new code as a sub-routine. In this case it's :$C083 C083 D392:20 17 03 N C080.

The addresses specified for these new features are for the corrected version of the Assembler - i.e. serial nos. greater than 1251; see note in AAL March '85. Here is a table of what to expect at each of the addresses used, so you can find the equivalent spots in other copies of the assembler:

```
$D198 -- 20 xx D2       (JSR GNC)
         B0 17          (BCS to RTS)
         49 30          (EOR #$30)

$D392 -- 20 xx D1       (JSR GET.VALUE)
         CA             (DEX)
         30 0E          (BMI to SEC)

$D40B -- 41 55 54 CF    (.AT /AUTO/)
         xx D3          (.DA AUTO-1)

$DB9A -- 09 80          (ORA #$80)
         9D 00 02       (STA $0200,X)
         C9 A0          (CMP #$A0)
```

Note that with the Auto/Manual Toggle function installed you won't need the MANUAL command any more, so you have a spare command if you need it!

```
                1000 *SAVE S.AUTO/MAN
                1010 *-------------------------------
                1020         .OR $300
                1030 *       .TF AUTO/MAN TOGGLE
                1040 *-------------------------------
5A-             1050 INCREMENT     .EQ $5A,5B
B8-             1060 SYM.VALUE     .EQ $B8,B9
E3-             1070 AUTO.FLAG     .EQ $E3
                1080
D028-           1090 WARM.START    .EQ $D028
D198-           1100 GET.VALUE     .EQ $D198
D40B-           1110 AUTO.CMD      .EQ $D40B
DB9A-           1120 INSTALL.CHAR  .EQ $DB9A
                1130 *-------------------------------
                1140 AUTO.MAN.CODE
0300- 8A        1150         TXA                  check cursor posn.
0301- F0 09     1160         BEQ .1               OK to output cmd.
0303- E0 07     1170         CPX #7               line start?
0305- B0 0F     1180         BGE .2               ignore ESC-U.
0307- 46 E3     1190         LSR AUTO.FLAG        cancel auto-mode.
0309- 4C 28 D0  1200         JMP WARM.START
                1210
030C- BD 0B D4  1220 .1      LDA AUTO.CMD,X       output cmd. name
030F- 20 9A DB  1230         JSR INSTALL.CHAR     put in buffer+scrn.
0312- E0 04     1240         CPX #4               4 chars. output?
0314- D0 F6     1250         BNE .1               no.
0316- 60        1260 .2      RTS                  exit ESC-U routine
                1270 *-------------------------------
                1280 *  Point start of AUTO cmd. handler
                1290 *  to here for extended function.
                1300 *-------------------------------
                1310 NEW.AUTO.EXT
0317- 20 98 D1  1320         JSR GET.VALUE        get linenum if any.
031A- 20 98 D1  1330         JSR GET.VALUE        get inc. if any.
031D- E0 03     1340         CPX #3               increment?
031F- 90 06     1350         BLT .1               no
0321- CA        1360         DEX                  adjust for inc.
0322- CA        1370         DEX                      do.
0323- A5 B8     1380         LDA SYM.VALUE        set inc. low byte
0325- 85 5A     1390         STA INCREMENT
                1400 *  (following 2 lines only needed
                1410 *   if you use increments of 255+1)
                1420 *       LDA SYM.VALUE+1     set inc. high byte
                1430 *       STA INCREMENT+1
0327- 60        1440 .1      RTS                  finish AUTO cmd.
```

12 Good Reasons Why RAMWORKS™ Is The Best Expansion Card For Your IIe

1 APPLEWORKS MEMORY Even though Ramworks enhances and expands a VAST ARRAY of other programs, Appleworks is our claim to fame. A 64K Ramworks will ADD 46K to your available desktop memory, a 128K Ramworks will ADD 91K, a 256K Ramworks will ADD 182K, and a 512K Ramworks will ADD 364K and a 1 meg Ramworks will give you nearly an 800K desktop. And it's all done automatically! When you plug in more memory chips into your Ramworks card, Appleworks will find them—automatically. Ramworks also increases the maximum number of records from 1350 to 4300.

2 APPLEWORKS SPEED AND POWER Ramworks does more than just increase the desktop memory (as if that weren't enough). With Ramworks, Appleworks will be able to run up to 20 times faster. If you buy a 256K or larger Ramworks card, Appleworks will automatically load itself in Ramworks. This greatly increases the speed at which Appleworks operates by eliminating all that nasty, time consuming disk access on Drive 1. These are but a few reasons why we say that Ramworks is Appleworks best friend.

3 EXPANDABILITY Ramworks was designed with the future in mind, as your needs increase, so can Ramworks. Clear instructions show you how to plug in more memory (up to 1 meg).

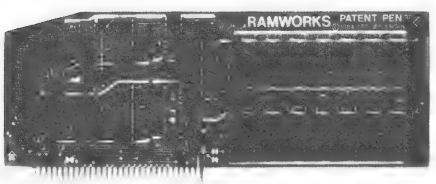
4 SPEED Today, as programs become more and more sophisticated, they inevitably become larger. And many of today's best selling programs (like Appleworks) won't fit in a 128K Apple, so many of these new larger programs continually go back to disk in search of more data. With Ramworks, you can have enough memory so that the entire program will be loaded into Ramworks' memory. This greatly increases the speed of software because your disk runs at 300 RPM, but Ramworks operates at the speed of light!

5 COLOR The same slot that's used for memory expansion is also the slot that's used for RGB color display, so all those lesser memory cards of yesterday make you decide in advance if you want RGB color. Only Ramworks lets you decide later to add RGB color. For only $129, an RGB option can be added to Ramworks to give you double high resolution color graphics and 80 column text. All with a razor sharp, vivid brilliance that's unsurpassed in the industry. The RGB option does not waste another valuable slot, but rather plugs into the back of Ramworks and attaches to any Apple compatible monitor. Remember, you can order the RGB option with your Ramworks or add it on at a later date.

6 COMPATIBILITY, OF THE SOFTWARE KIND Programs like Appleworks, Magic Office System, Flashcalc, The Spread Sheet, Diverse-A-Dos, Supercalc, Magicalc and many others automatically recognize all or most of Ramworks memory (512K is average). The simple fact is that Ramworks is compatible with more off-the-shelf software than any other RAM card. Ramworks is 100% compatible with ALL software written for the Apple 80 column and extended 80 column card. Additionally, Ramworks can emulate other RAM cards so software written for other cards will run without modification. Software written for RAMWORKS will not work on other cards. We can emulate others, but others can't emulate us.

7 COMPATIBILITY, OF THE HARDWARE KIND Unlike others, Ramworks is fully compatible with hardware add on's from other companies, like the Sider and Profile hard disks. And Ramworks was designed in accordance with the official expansion rules defined by Apple so you don't have to worry about compatibility problems. As you continue to expand and make your Apple more powerful with other expansion products from Applied Engineering, you'll appreciate how each product has extra features designed to work with Ramworks and other products to give you a total performance package that is more powerful than the sum of its parts.

8 IT SELLS THE MOST Popularity translates into great software support because software companies can't support all RAM cards, they can only support the ones their customers are likely to own. And software companies appreciate the fact that when they write software for Ramworks in the IIe, they're also writing software for our memory expansion card for the IIc, Z-RAM. And our customer list reads like the Who's Who of Apple computing with just about every software company in the land buying one, including Apple Computer (in the hundreds), Rupert Lissner, and Steve Wozniak (we didn't give one to Mr. Wozniak just to use his name, 2 one meg Ramworks were paid for at full price).



9 IT'S FROM APPLIED ENGINEERING Unlike most of the competition, we only make accessories for Apple, so we'll never spend your money on IBM product research. Applied Engineering's years of experience and wide product line really pays off, and because of our high sales levels we buy most of our I.C. chips factory direct. So don't let our low prices fool you, they're caused by high volume production. That's why we can offer the most memory for the least money. Guaranteed!

10 IT'S GOT IT ALL

- ☑ Sharp 80 Column Text
- ☑ Double high resolution graphics (with or without RGB option)
- ☑ User Expandable to 1 Megabyte
- ☑ Can Use 64K or 256K RAMS in any combination
- ☑ Adds Memory to Appleworks
- ☑ Accelerates Appleworks
- ☑ 100% Compatibility with All IIe software
- ☑ RAM Disk software available, compatible with Applesoft, PRO-DOS, DOS 3.3, and PASCAL ($29)
- ☑ RAM Disk available for CP/M ($29). (This program is included with our CP/M card)
- ☑ Visicalc preboot available ($29)
- ☑ RGB option
- ☑ Takes only one slot
- ☑ 3 year no hassle warranty

11 THE PATENT OFFICE HAS ONE There are many advanced features on Ramworks, but two parts of the design are so advanced we applied for patents. One patent application deals with our ultra fast, ultra smooth 80 column screen display, and the other patent application deals with our ingenious way of dramatically reducing the power and heat of memory chips and improving reliability at the same time.

12 HERE TODAY, HERE TOMORROW In the seven years we've been making products for the Apple, we've seen a lot of companies come and go. Although nothing is forever, we're growing, expanding and we're profitable. And we are totally committed to Apple computing, which means you'll never run out of things to do with Ramworks. Or for that matter, reasons to buy one.

Ramworks™ with 64K	**$179**
Ramworks™ with 128K	**$249**
Ramworks™ with 256K	**$299**
Ramworks™ with 512K	**$399**
Ramworks™ with 1 MEG	**$649**
RGB Option (can be added later)	**$129**

Call (214) 241-6060
9 a.m. to 11 p.m. 7 days a week or send check or money order to:
Applied Engineering, P. O. Box 798, Carrollton, Texas 75006

MasterCard VISA Visa and
C.O.D. welcome. No extra charge for credit cards. Texas Residents add 5¼% sales tax. Add $10.00 if outside U.S.A.

APPLIED ENGINEERING
"We Set the Standard"

Apple ProDOS: Advanced Features for Programmers, a Review.....

.....Bill Morgan

Gary Little, the prolific author of Inside the Apple //e and Inside the Apple //c, has yet another new book out. This one is called Apple ProDOS: Advanced Features for Programmers. In this volume Little covers just about all you need to know to write assembly language programs under ProDOS, from simply passing commands to BASIC.SYSTEM, through great detail on all the MLI calls, to writing your own interrupt handlers and device drivers.

Here's a quick summary of the book's contents:

1 - An Introduction to ProDOS -- Little starts out with the history of Apple's DOS's, a comparison of ProDOS and DOS 3 3 and a summary of important features of ProDOS.

2 - Files and File Management -- Here he covers the directory structures, file structures, disk formatting, and gives us a READ.BLOCK program.

3 - Loading and Installing ProDOS -- This chapter goes into the boot process, ProDOS' memory usage, and the Global Page.

4 - The Machine Language Interface -- This is the information on using the MLI, its error codes, and complete details of all MLI calls.

5 - System Programming Featuring BASIC.SYSTEM -- Here we have a discussion of system programs, the structure and commands of BASIC.SYSTEM, and assembly language programming under BASIC.SYSTEM.

6 - Interrupts -- In this chapter Little covers interrupts in general, ProDOS interrupt handling, and programming the Apple mouse.

7 - Disk Drivers -- Nearing the end, we go into identifying and handling foreign disk drivers, driver commands, the /RAM driver, and adding your own driver

8 - ProDOS Clock Drivers -- And finally we find out about using the built-in clock support, adding a clock driver, and reading the date and time from Applesoft.

An important strength of this book is the wealth of examples. In the chapter on the Machine Language Interface there is an example of the correct use of EVERY MLI call. The BASIC.SYSTEM chapter includes an ONLINE command, to identify all disk volumes currently on line. The chapter on interrupts contains a couple of examples of mouse programming. The Disk driver section has a listing of a simple /RAM driver for main memory. And this is just a sample of the useful code provided in Little's new book. A companion disk containing all of the book's programs and more is available for $25.00 from the author.

I hear some of you asking: How does Apple ProDOS: Advanced Features (APAF) compare to Beneath Apple ProDOS (BAP)? Well, the two books complement each other quite nicely. With all its examples, treatment of interrupt handlers and device drivers, and overall clarity, I'd say that APAF is the better book on programming under ProDOS BAP has useful examples as well, and better detail about the internals of diskette formatting and how ProDOS works, especially with its 120+ page supplement describing the code on a line-by-line basis. So if you're concerned with understanding the inner workings of the operating system, or with modifying its behavior, BAP is the book to have. Otherwise, get APAF for the best information on programming using ProDOS. Personally, I'm glad to have both books on the shelf here, along with Apple's ProDOS Technical Reference Manual.

Apple ProDOS: Advanced Features for Programmers, by Gary B. Little. Brady Communications Co., 1985. 266+iv pp., Reference Card. $17.95. Available from S-C Software for $17 + shipping.

Adapting the Output Format of Rak-Ware DISASM.....Bob Kovacs

This technical note describes the format table used within DISASM 2.2e, which can be modified to adapt the output text file format to other assemblers. Even if you never plan to modify DISASM, or even if you don't own a copy of DISASM, you can learn a lot about the use of configuration tables by studying what follows.

The current version of the disassembler provides three different output formats to support the DOS ToolKit, S-C, and LISA assemblers. The format table contains various attributes which are unique to each assembler. The table begins at location $1331 and is $3F bytes long. Let's first examine the table and then determine how to adapt it to other assembler formats.

Item	ToolKit		S-C		LISA	
comment	AA	*	AA	*	BB	;
firstchr	00	none	89	^I	00	none
tabchrl	A0	spc	89	^I	A0	spc
tabchr2	A0	spc	A0	spc	A0	spc
opchr	C1	A	00	none	00	none
pgzchr	C5D1D5	EQU	AEC5D1	.EQ	C5D0DA	EPZ
extchr	C5D1D5	EQU	AEC5D1	.EQ	C5D1D5	EQU
hexchr	C4C6C2	DFB	AEC8D3	.HS	C8C5D8	HEX
orgchr	CFD2C7	ORG	AECFD2	.OR	CFD2C7	ORG
prechr	AA0000	*	000000	none	C9CED3	INS
postchr	00	none	98	^X	85	^E

comment: the character used at the beginning of a line to signify a comment line.

firstchr: the character ouput at the beginning of each line.

tabchrl: the character used to tab to the opcode field.

tabchr2: the character used to tab to the operand field.

opchr: operand for impled accumulator instructions (ASL, LSR, ROR, ROL).

pgzchr: directive for page zero declarations.

extchr: directive for absolute declarations.

hexchr: directive for data tables.

orgchr: directive for setting the program origin.

prechr: preamble sequence for initialization of the assembler

postchr: postamble character for termination of the assembler's loading operation.

You will find that it is relatively simple to modify the format table for other assemblers. First, determine which of the three existing formats is to be overwritten (just pick the one you think you'll need the least). Then determine the format data which is appropriate to your assembler BLOAD DISASM, enter the monitor, and stuff the new values into the table. Finally BSAVE DISASM,A$800,L$D00.

Or, if you have purchased the source code of DISASM 2.2e (or created your own using DISASM!), you can merely edit the table with your assembler and re-assemble the program.

You might also need or want to change some other parameters, which are not in the format table:

Label Prefix: located at $132E, the current value is C9DAD8 (the letters "IZX"). These letters are used to indicate internal, pagezero, and external labels in the generated text file.

Menu Table: located at $1300, this table contains the names of the three assemblers listed in the first menu. Each name is stored in ASCII, followed by a return ($0D) and a terminator ($00).

Label Name Separator: A period ($AE) is output as the second character in every generated label name. This can be changed to any other character by editing the LDA #$AE instruction at location $0EA4.

I would be interested in hearing from any of you who have already modified DISASM. This kind of feedback can lead to new versions with even more powerful features.

DATE Command for ProDOS.....Bill Morgan

One of the nice new features in ProDOS is the way the diskette catalog shows the date of creation and last modification for each file, IF you have a clock/calendar card installed in your Apple. Well I don't have such a card in either of the Apples I use regularly, at work or at home. And no //c has a clock! (Yet, at least. I'll bet someone will come up with a way...)

Anyway, I got tired of always seeing <NO DATE> and started figuring out how to set a date without a clock to do it for me. A look at Beneath Apple ProDOS informed me that the current date is transformed into the format YYYYYYYMMMMDDDDD and stored (in the usual 6502 low byte/high byte sequence) at $BF90-BF91 in the ProDOS Global Pages (the fixed locations of all of the accessible system variables). The first thing I did was manually convert the current date into that format and poke it in from the Monitor. That went like this:

```
                                  $BF90      $BF91
May =  $5 = 0101             MMM DDDDD  YYYYYYY M
 10 =  $A = 01010            101 01010  1010101 0
'85 = $55 = 1010101             $AA        $AA
```

So, the values to poke into $BF90-91 were $AA and $AA. What better time than a four-A day to start such a project!

That experiment worked just fine: the next file I saved on the disk showed creation and modification dates of 10-MAY-85, just as I had hoped. With that success under my belt the next step had to be to come up with a program to read and/or set those date bytes. And, while I'm at it, why not take advantage of ProDOS' built-in hooks for installing new commands and add a DATE command to the operating system?

How do I go about adding a command? The ProDOS Technical Reference Manual is pretty sketchy on the subject, but two other books, Beneath Apple ProDOS and the new Apple ProDOS: Advanced Features for Programmers, have good descriptions and examples of the procedure. If you're going to do much assembly language programming under ProDOS you should have one or both of those books.

When ProDOS fails to recognize a command it does a JSR EXTRNCMD ($BE06) to find out if an external command processor will claim this one. What I have to do is install the address of DATE in $BE07-08, after moving the address that was already there into a JMP instruction. This way, if DATE doesn't recognize the command it can pass it along to any other processor that might have been there before.

Processing of an external command is normally divided into two phases, a parser and a handler. The parser section will scan the command name at the beginning of the line. If the command is not recognized, the parser should set the carry bit and JMP on to the address found in EXTRNCMD to see if another external processor will claim it.

If the command is recognized, the parser can set certain bits in PBITS ($BE54-55) to signify which parameters are permitted or required on the command line, and store the address of the handler in EXTRNADDR ($BE50-51). After storing the command length minus one in XLEN ($BE52) and a zero in XCNUM ($BE53), to signify that an external processor did claim the command, the parser then returns control to ProDOS to scan the rest of the line. If the line was syntactically correct, ProDOS will return the values of the parameters in a set of standard locations ($BE58-6F) and pass control back to the handler address specified.

Since DATE is a simple processor that uses a nonstandard parameter, I just set PBITS to zero, to indicate no parsing necessary, and store the address of an RTS instruction in EXTRNADDR. I then proceed to do all my processing before returning to ProDOS

There is one additional wrinkle to using an external command with ProDOS: where do I put my code so ProDOS, Applesoft, and others don't stomp all over it? In the interest of simplicity I have ignored that problem here. The best procedure, as shown in the books mentioned above, is to call ProDOS to assign me a buffer and then relocate my code into that buffer. The examples in the books provide details of this process.

Now, let's take a look at the code:

Lines 1310-1400 install DATE by moving the current External Command address to my exit JMP instruction and storing DATE's address in the vector.

Lines 1440-1540 check the input buffer to see if this is a DATE command. If not we branch on down to that JMP instruction where we earlier put the address found in the External Command vector. This passes control either on to the next external command in the chain, or back to ProDOS for a SYNTAX ERROR.

If the command matched we go on to lines 1560-1650 to do the necessary housekeeping. This involves storing the command length-1 in the Global Page, setting a couple of flags to tell ProDOS not to parse the rest of the command line, and that an external command has taken over. Then we supply a handler address for the second half of ProDOS' processing, which in this case is just an RTS instruction. Finally we reach lines 1670-1690, where we check to see if the character following DATE is a Carriage Return. If so we branch forward to RETURN.DATE to display the existing date.

If there is more than just DATE on the command line, we must want to set a new date, so we fall into SET.DATE at line 1710. This routine makes heavy use of ACCUMULATE.DIGITS at line 2400, so we'll examine that code first. The first step is to zero the byte where we'll be accumulating the value typed in. Then we scan forward in the input buffer, looking for a nonblank character. When we find one we first check to see if it is a slash, which marks the end of a number, or a Carriage Return, which marks the end of the line. If it was either of those we

exit, setting the Carry bit to indicate which one we found.

If the character found was not a delimiter we next check to see if it is a number. If not, we have a SYNTAX ERROR. When we do get a number, we strip off the high bits to convert the ASCII code to a binary value, and save that value. We then multiply the previous value in ACCUM by 10 and add in the new value. Then it's back to line 2440 to get another character. Lines 2710-2730 load the A-register with the value found and branch to the error exit if that value was zero.

Now, back to SET.DATE. That routine begins at line 1720 with a DEY to get ready for the INY at the beginning of ACCUMULATE.DIGITS. We then get the month, check for a legal value, and store it. Next we get the day, save the status, and check and save that value. Then it's time to check the status to see if the day was followed by a slash, or by a Carriage Return. If it was a slash then a year was specified, so we go get that value. If it was a Return no year was present, so we use 1985. (I guess that means we'll have to reassemble or patch this program every year. I think I can handle that.)

The last step in SET.DATE is to fold the year, month, and day together as described above and store the results in the Global Page. The comments in the listing illustrate how the bits are shuffled around to the correct format. After setting the date we fall into RETURN.DATE to display the result.

RETURN.DATE, at lines 2080-2290, is quite straightforward. It just gets the bytes from the Global Page, unfolds them, and calls DEC.OUT to translate them to decimal numbers and display those numbers. Again, the comments illustrate the bit manipulations involved in the unfolding process.

The final section of code is DEC.OUT, at lines 2750-2910. In lines 2760-2810 we use the Y-register to count how many times we can subtract 10 from the number passed in the A-register. Then lines 2830-2910 restore and save the A-register, make sure the tens count is non-zero, convert it to a character and print it. We then recover the units value and print that out.

```
                1000 *SAVE S.DATE
                1010 *--------------------------------
                1020 *
                1030 *       Program to read or set the date
                1040 *       bytes in the ProDOS Global Page
                1050 *
                1060 *             by Bill Morgan
                1070 *
                1080 *--------------------------------
40-             1090 POINTER     .EQ $40,41
42-             1100 ACCUM       .EQ $42
43-             1110 MONTH       .EQ $43
44-             1120 DAY         .EQ $44
45-             1130 TEMP        .EQ $45
                1140
0200-           1150 WBUF        .EQ $200
                1160
BE07-           1170 EXTRNCMD    .EQ $BE07
BE50-           1180 EXTRNADDR   .EQ $BE50,51
BE52-           1190 XLEN        .EQ $BE52
BE53-           1200 XCNUM       .EQ $BE53
BE54-           1210 PBITS       .EQ $BE54
BF90-           1220 GP.DATE     .EQ $BF90
```

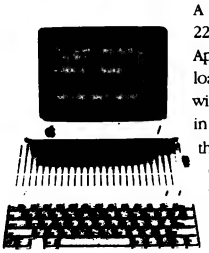
Expanding Your IIc Is Easy With Z-RAM

Applied Engineering and Apple computer have teamed up to take your IIc to new heights.

Applied Engineering's Z-RAM card for the IIc is available with 256K or 512K of additional memory and a powerful Z-80 microprocessor for running CP/M software.

Z-RAM fits neatly inside the IIc. Installation is easy, clear instructions show you how. You'll need a screwdriver and about 10 minutes (if you can change a light bulb you can install Z-RAM).

Z-RAM and Appleworks will knock your socks off.



A 256K Z-RAM will give you a 229K available desktop and Appleworks will be completely loaded into memory. Appleworks will now run about 10 times faster in your IIc with 1 disk drive than in other IIc's with 2 disk drives. A 512K Z-RAM will give you a 413K available desktop. A 256K Z-RAM can be upgraded to 512K by just plugging in more memory chips.

Z-RAM is also a high speed solid state disk drive. With Z-RAM, your programs will load and save over 20 times faster. Z-RAM's RAM disk is compatible with Applesoft, ProDOS, DOS 3.3, PASCAL and CP/M. And with Z-RAM, you can copy a disk in one pass. Just insert the original, remove the original, insert blank disk! That's it! Z-RAM is another disk drive, only 20 times faster, 4 times larger capacity, and no whirring, clicking or waiting!

But before you start panting over all that extra memory, don't forget that the Z-RAM card has a built-in high speed Z-80 processor chip that allows you to run CP/M programs like Wordstar, dBASE II, Turbo PASCAL, Microsoft BASIC, FORTRAN and COBOL and over 3,000 other CP/M programs. So Z-RAM not only makes Apple programs run better and faster, it lets you run MORE programs.

With the Z-RAM card installed, your IIc is still your IIc only now you'll have that extra memory that Appleworks and other programs need. And you can run all that great CP/M software that others can only dream about.

Z-RAM is 100% compatible with all IIc software and hardware including the mouse, 2nd disk, modem and printer. Z-RAM is easily handled by the IIc power supply as power consumption is kept very low by using two custom integrated circuits and a patent pending power saving design. And Z-RAM is from Applied Engineering, the acknowledged leader and innovator of accessories for the Apple.

Z-RAM comes complete with manual, RAM disk software, Z-80 operating system, CP/M manual and a 3 year no hassle warranty.

So the next time somebody asks you why you didn't get an IBM P.C., tell him you bought a IIc because the IBM didn't have enough memory and was too slow and couldn't run CP/M software. And tell him you made it past the 8th grade.

Z-RAM with 256K **$449**
Z-RAM with 512K **$549**

If you want to run CP/M software, but don't need more memory, may we suggest our Z-80c card. The Z-80c offers the same CP/M performance as Z-RAM but has no memory expansion ports. And the Z-80c will not affect the running of Apple programs. The Z-80c is priced at only $159.00 and should you ever want to upgrade to Z-RAM, we'll refund your full purchase price.

Call (214) 241-6060
9 a.m. to 11 p.m. 7 days a week or

Send check or money order to:
Applied Engineering
P. O. Box 798
Carrollton, Texas 75006

MasterCard Visa and C.O.D. welcome. No extra charge for credit cards. Texas residents add 5⅛% sales tax. Add $10.00 if outside U.S.A.

APPLIED ENGINEERING
"We Set the Standard"

```
                        1230
F941-                   1240 PRAX       .EQ $F941
FD8E-                   1250 CROUT      .EQ $FD8E
FDED-                   1260 COUT       .EQ $FDED
                        1270 *--------------------------------
                        1280         .OR $803
                        1290 *       .TF B.DATE
                        1300 *--------------------------------
                        1310 INSTALL
0803- AD 08 BE          1320         LDA EXTRNCMD+1     exit to old
0806- 8D BA 08          1330         STA EXIT+2         user command
0809- AD 07 BE          1340         LDA EXTRNCMD
080C- 8D B9 08          1350         STA EXIT+1
080F- A9 08             1360         LDA /DATE          become new
0811- 8D 08 BE          1370         STA EXTRNCMD+1     user command
0814- A9 1E             1380         LDA #DATE
0816- 8D 07 BE          1390         STA EXTRNCMD
0819- 60                1400         RTS
                        1410 *--------------------------------
081A- 44 41 54
081D- 45                1420 COMMAND .AS /DATE/
                        1430 *--------------------------------
081E- A0 00             1440 DATE    LDY #0
0820- 84 40             1450         STY POINTER        point to input buffer
0822- A9 02             1460         LDA /WBUF
0824- 85 41             1470         STA POINTER+1
0826- B1 40             1480 .1      LDA (POINTER),Y    scan command
0828- 29 7F             1490         AND #%01111111
082A- D9 1A 08          1500         CMP COMMAND,Y
082D- D0 35             1510         BNE ERR.BRIDGE     not mine
082F- C8                1520         INY
0830- C0 04             1530         CPY #4
0832- 90 F2             1540         BCC .1
                        1550 *--- ProDOS bookkeeping ---------
0834- 88                1560         DEY
0835- 8C 52 BE          1570         STY XLEN           command length - 1
0838- C8                1580         INY
0839- A9 00             1590         LDA #0
083B- 8D 54 BE          1600         STA PBITS          don't parse parms
083E- 8D 53 BE          1610         STA XCNUM          external command
0841- A9 B4             1620         LDA #RTS1
0843- 8D 50 BE          1630         STA EXTRNADDR      no execution after
0846- A9 08             1640         LDA /RTS1          command parsing
0848- 8D 51 BE          1650         STA EXTRNADDR+1
                        1660 *--- set or display date? -------
084B- B1 40             1670         LDA (POINTER),Y
084D- C9 8D             1680         CMP #$8D           DATE only?
084F- F0 3A             1690         BEQ RETURN.DATE    yes, return old date
                        1700 *--------------------------------
                        1710 SET.DATE
0851- 88                1720         DEY
0852- 20 BB 08          1730         JSR ACCUMULATE.DIGITS  get month
0855- C9 0D             1740         CMP #13
0857- B0 5E             1750         BCS ERROR          >12 no good
0859- 85 43             1760         STA MONTH
085B- 20 BB 08          1770         JSR ACCUMULATE.DIGITS  get day
085E- 08                1780         PHP                save status
085F- C9 20             1790         CMP #32
0861- 90 03             1800         BCC GO.ON          <=31 ok
                        1810
0863- 28                1820         PLP
                        1830 ERR.BRIDGE
0864- D0 51             1840         BNE ERROR       ...always
                        1850
0866- 85 44             1860 GO.ON   STA DAY
0868- 28                1870         PLP                recover status
0869- 90 04             1880         BCC .1             .CC. if "/"
086B- A9 55             1890         LDA #85            year defaults to '85
086D- D0 07             1900         BNE .2          ...always
086F- 20 BB 08          1910 .1      JSR ACCUMULATE.DIGITS  get year
0872- C9 64             1920         CMP #100
0874- B0 41             1930         BCS ERROR          >99 no good
0876- 48                1940 .2      PHA                save year
0877- A5 43             1950         LDA MONTH          X 0000MMMM
0879- 4A                1960         LSR                M 00000MMM
087A- 6A                1970         ROR                M M00000MM
087B- 6A                1980         ROR                M MM00000M
087C- 6A                1990         ROR                M MMM00000
```

```
087D- 85 43     2000          STA MONTH
087F- 68        2010          PLA                    M OYYYYYYY
0880- 2A        2020          ROL                    O YYYYYYYM
0881- 8D 91 BF  2030          STA GP.DATE+1
0884- A5 43     2040          LDA MONTH                MMMOOOOO
0886- 05 44     2050          ORA DAY                  MMMDDDDD
0888- 8D 90 BF  2060          STA GP.DATE
                2070 *--------------------------------
                2080 RETURN.DATE
088B- 20 8E FD  2090          JSR CROUT
088E- AD 91 BF  2100          LDA GP.DATE+1          X YYYYYYYM
0891- 4A        2110          LSR                    M OYYYYYYY
0892- 48        2120          PHA
0893- AD 90 BF  2130          LDA GP.DATE            M MMMDDDDD
0896- 48        2140          PHA
0897- 6A        2150          ROR                    X MMMMDDDD
0898- 4A        2160          LSR                    X OMMMMDDD
0899- 4A        2170          LSR                    X OOMMMMDD
089A- 4A        2180          LSR                    X OOOMMMMD
089B- 4A        2190          LSR                    X OOOOMMMM
089C- 20 F1 08  2200          JSR DEC.OUT            display month
089F- A9 AF     2210          LDA #"/"               /
08A1- 20 ED FD  2220          JSR COUT
08A4- 68        2230          PLA                    X MMMDDDDD
08A5- 29 1F     2240          AND #%00011111         X OOODDDDD
08A7- 20 F1 08  2250          JSR DEC.OUT            display day
08AA- A9 AF     2260          LDA #"/"               /
08AC- 20 ED FD  2270          JSR COUT
08AF- 68        2280          PLA                    X OYYYYYYY
08B0- 20 F1 08  2290          JSR DEC.OUT            display year
                2300 *--------------------------------
                2310 GOOD.EXIT
08B3- 18        2320          CLC                    signal no error
08B4- 60        2330 RTS1     RTS
                2340 *--------------------------------
08B5- 68        2350 ERROR1   PLA                    clean up
08B6- 68        2360          PLA                    return addresses
08B7- 38        2370 ERROR    SEC                    signal error
08B8- 4C B4 08  2380 EXIT     JMP RTS1               INSTALL makes address
                2390 *--------------------------------
                2400 ACCUMULATE.DIGITS
08BB- A9 00     2410          LDA #0
08BD- 85 42     2420          STA ACCUM              zero accumulator
                2430
08BF- C8        2440 .1       INY                    next character
08C0- B1 40     2450          LDA (POINTER),Y
08C2- 29 7F     2460          AND #%01111111         hi-bit off
08C4- C9 20     2470          CMP #' '               space?
08C6- F0 F7     2480          BEQ .1                 back for another
08C8- C9 2F     2490          CMP #'/'               slash?
08CA- F0 1F     2500          BEQ .2                 yes, exit .CC.
08CC- C9 0D     2510          CMP #$0D               <CR>?
08CE- F0 1C     2520          BEQ .3                 yes, exit .CS.
08D0- C9 30     2530          CMP #'0'               too small?
08D2- 90 E1     2540          BCC ERROR1             not digit
08D4- C9 3A     2550          CMP #'9'+1             too big?
08D6- B0 DD     2560          BCS ERROR1             not digit
                2570
08D8- 29 0F     2580          AND #%00001111         isolate value
08DA- 85 45     2590          STA TEMP               stash it
08DC- A5 42     2600          LDA ACCUM
08DE- 0A        2610          ASL                    X 2
08DF- 0A        2620          ASL                    X 4
08E0- 65 42     2630          ADC ACCUM              X 5
08E2- 0A        2640          ASL                    X 10
08E3- 65 45     2650          ADC TEMP               add new digit
08E5- B0 CE     2660          BCS ERROR1             too big
08E7- 85 42     2670          STA ACCUM
08E9- 90 D4     2680          BCC .1           ...always
                2690
08EB- 18        2700 .2       CLC                    .CC. if /
08EC- A5 42     2710 .3       LDA ACCUM              return value
08EE- F0 C5     2720          BEQ ERROR1             0 no good
08F0- 60        2730          RTS
                2740 *--------------------------------
```

```
                    2750 DEC.OUT
08F1- A0 00         2760         LDY #0          zero counter
08F3- 38            2770         SEC             get ready
08F4- E9 0A         2780 .1      SBC #10         subtract 10
08F6- 90 03         2790         BCC .2          borrow?
08F8- C8            2800         INY             count a 10
08F9- 10 F9         2810         BPL .1       ...always
                    2820
08FB- 69 0A         2830 .2      ADC #10         restore borrow
08FD- 48            2840         PHA             save units
08FE- 98            2850         TYA             print 10's count
08FF- F0 05         2860         BEQ .3          no leading zero
0901- 09 B0         2870         ORA #$B0        make character
0903- 20 ED FD      2880         JSR COUT        print it
0906- 68            2890 .3      PLA             recover units
0907- 09 B0         2900         ORA #$B0        make character
0909- 4C ED FD      2910         JMP COUT        return through COUT
```

32-bit Values in Version 2.0 -- A Mixed Blessing.....Bob S-C

In previous versions of the S-C assemblers, expressions were evaluated in 16 bits, and symbol values were kept in the table in 16-bit form. Version 2.0 works with 32-bit expressions and symbol values. We added this feature for your benefit. but it may sometimes be a mixed blessing.

For example, Bob Bernard had a problem with a program which assembled perfectly under Version 1 0. but gave countless BAD ADDRESS errors in version 2.0. We traced the problem to his origin statement. which was ".OR -31488". In older versions, -31488 is the same as $8500, but in version 2.0 it is $FFFF8500. The following code will not assemble:

```
      .OR -31488
SSS   JMP SSS
```

Why? Because the value of SSS is also $FFFF8500, and it will not fit in a JMP instruction. In 65816 mode, using a JML instruction, it would be legal.

Two ways to fix come to mind. You normally work in hexadecimal when you are in assembly language, rather than decimal. Therefore, change the origin statement to ".OR $8500". Or. if you really want to use decimal, write ".OR 65536-31488".

Another owner of version 2.0 had a problem with a program that used many macros, and lots of private labels. Private labels are the ones used inside macro definitions, which are written with a colon and a one or two digit number. The private label table normally begins at $FFF and grows downward toward $800. His program assembled with no problems before, but under version 2.0 it got a MEM FULL error Reason, again, the 32-bit symbol values. Each entry in the private label table now takes two more bytes, so he ran out of space sooner. His solution was to move the beginning of the label table higher.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $18 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada. and Mexico; add $14 postage for other countries. Back issues are available for $1.80 each (other countries add $1 per back issue for postage).